A Practical Path-planning Algorithm for a Vehicle with a Constrained Turning Radius: a Hamilton-Jacobi Approach

Ryo Takei[1], Richard Tsai[2], Haochong Shen[3], and Yanina Landa[1]

***Abstract*— We consider the problem of optimal path planning of a forward moving *simple car* with a minimum turning radius (a Dubins' car), in a known environment possibly containing inpenetrable obstacles. We model this problem in the Hamilton-Jacobi partial differential equation framework, and provide a finite difference numerical method to solve it. Furthermore, we introduce and demonstrate a scheme based on this framework to steer a micro-car on a testbed [29].**

I. Introduction

We consider the problem of generating the shortest curve from an initial position $\mathbf{x}_0$ and a tangent direction θ_0 to a target position $\mathbf{x}_f$ (and possibly also a tangent direction θ_f), with a lower bound on the curvature. This problem arises when optimally steering a *simple car* – a vehicle with a fixed rear axle and a pair of turning front wheels, as in Fig. 1. If the wheels have a maximum turning angle constraint, $|\phi| \leq \phi_{\max}$, it can be shown that, equivalently, the curvature of a forward-moving path is bounded below by $\rho_{\min} = L/\tan(\phi_{\max})$, where L is the distance between the front and rear axles [1, Chapter 13]. Steering such a vehicle is a non-trivial task, especially in the presence of obstacles and/or with the freedom of reversing direction.

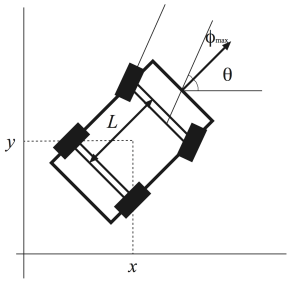

Fig. 1. A simple car.

A. Previous Work for Curvature Constrained Path Planning

Traditionally, combinatorial methods based on path geometry have been the primary tools for constrained path-planning problems. The problem of finding the shortest path under the curvature contraint was first introduced in the pioneering work of Dubins [2], where he characterized such paths in two dimensions in the absence of obstacles. In the robotics community, a *Dubins' car* refers to a vehicle with a minimum turning radius that only moves forward. Reeds and Shepp [3] investigated its variant (still in the absence of obstacles), called the *Reeds-Shepp's car*, allowing for reversal. Both Dubins' and Reeds-Shepp's cars have optimal paths that could be classified into known permutations of a sequence consisting of *bang-bang controls*; that is, traveling straight, turning fully right or fully left, and, for the latter car, reversing direction. Reachability problems of such cars have been studied in [4], [5].

There are several approaches for computing curvature-constrained optimal paths. Barraquand and Latombe [6] considered creating a reachability tree assuming that, for example, a Dubins' car would locally only move straight, fully right or fully left for a short distance. The tree is grown from the target point backwards until one of the leaves reaches the starting point. Unnecessary cluttering of leaves can be avoided by partitioning the domain into cells and restricting to one leaf per cell [1, Chapter 14].

In the presence of obstacles, global approaches for the Dubins' car problem have extended the analysis of characterizing path geometries. For n polygonal obstacles, Wang and Agarwal [7] gave a $O((n^2/\epsilon^2)\log n)$ algorithm to construct a path that is no longer than $(1+\epsilon)$ times the shortest ϵ-*robust* path. Informally, a path is ϵ-robust if pertubations by $\epsilon/2$ is still feasible. For similar results for a more general class of obstacles see [8] and [9].

Jacobs and Canny [10] presented a grid search algorithm (also computing ϵ-robust paths), where certain nodes are strategically placed along the obstacle edges. Their setup is somewhat similar to the one proposed in this paper in that each node represents both the position and orientation of the vehicle. Two nodes are adjacent if there is a collision free trajectory connecting them. Then Dijkstra's algorithm [11] is applied to find the shortest path among ordered sequences of adjacent nodes.

Reif and Wang [12] developed a non-uniform grid algorithm in higher dimensions using the obstacle-free optimal paths as building blocks to construct optimal paths among obstacles. Lee et. al. [13] considered the problem of finding an optimal curvature-bounded path that passes through a sequence of way-points known *a priori*. For the Reeds-Shepp's car, a two phase algorithm of Laumond and Jacobs [14] first finds a feasible path ignoring curvature constraints, then modifies the path to satisfy them; the method is efficient, but the resulting paths need not be optimal.

Our approach is very similar to that of [15]: first determine a cost-to-go function (the *value function* (6)), then use it to compute individual trajectories. In [15], the cost-to-go func-

[1]Dept. of Mathematics, University of California Los Angeles, Los Angeles, CA 90095 {rtakei, ylanda}@math.ucla.edu
[2]Dept. of Electrical Engineering, University of California Los Angeles, Los Angeles, CA 90095 hshen@ee.ucla.edu
[3]Dept. of Mathematics, University of Texas at Austin, Austin, TX 78704 ytsai@math.utexas.edu

tion is determined by discretizing the *dynamic programming principle* (7) on tetrahedrons constructed using barycentric coordinates. Instead, we propose to discretize a related partial differential equation (PDE). Thus, we can exploit the well-developed theory of such PDEs and the corresponding robust and simple-to-implement numerical methods.

B. The Hamilton-Jacobi Formulation for Path Planning

In comparison to traditional methods, we introduce a PDE characterization of the problem: a Hamilton-Jacobi (HJ) equation.

The HJ equation is a first order non-linear PDE that has the general form [16]

$$H(\nabla u(\mathbf{x}), \mathbf{x}) = r(\mathbf{x}). \tag{1}$$

One of the most commonly known HJ formulation for path planning is the *eikonal equation*, where $H(\nabla u(\mathbf{x}), \mathbf{x}) = |\nabla u(\mathbf{x})|$ and $r(\mathbf{x}) = 1$. Coupled with a boundary condition $u|_\Gamma = 0$, it encodes the distance to the set Γ. An important property of the eikonal equation is that its characteristic curves coincide with optimal paths.

The underlying concept in the derivation of the eikonal equation and other optimal control problems is the celebrated *dynamic programming principle* (DPP) [17]. We will illustrate this when deriving our HJ equation in section II-B. Note that, while the DPP can be used to derive an algorithm, the HJ equation gives a closed-form solution to the problem. For a mathematical treatment of DPP and HJ equations, see [19], [18].

To numerically solve the eikonal equation (and more general HJ equations), we mention the idea introduced in [20]: from a grid node, the scheme is derived by tracing the characteristics in the upwind direction and reconstructing the solution value at the nearest intersection of the characteristics and the grid by suitable interpolation using neighboring grid nodes. Similar ideas are used for the fast marching method [21], [22], which is a variant of Dijkstra's algorithm, and the fast sweeping method [23], [24]. In contrast, in the method of reachability trees [6], the paths are globally traced out ignoring grid nodes. For numerical methods for more general HJ equations, see [25], [26], [27].

C. Paper Overview

The main contribution of this article is two-folds. First, in section II-B, we formally derive a HJ equation for the Dubins' car problem. To the authors' knowledge, this approach is new. In sections II-C and II-E, we provide a numerical discretization to solve the HJ equation and show numerical results.

Secondly, in section III-A, we introduce a practical scheme for steering an actual vehicle in a bounded domain with obstacles. We argue that the aforementioned HJ formulation is particularly suited for such a problem. In section III-C, the performance of this algorithm is validated in a testbed setting.

II. The Hamilton-Jacobi Formulation

A. Definitions and Setup

Let $\Omega \subset \mathbb{R}^2$ be a bounded, connected domain, which we call a *map*. The map is partitioned into its *free space* and *obstacles*: $\Omega = \Omega_{free} \cup \Omega_{obs}$. We will denote by $\mathbf{x} = (x, y) \in \Omega$ a point (the *location*) on the map. Let the *pose* be the ordered triple of the location and orientation (in radians) of the vehicle: $(\mathbf{x}, \theta) = (x, y, \theta) \in \Omega \times [0, 2\pi)$. Define a *path* or a *trajectory* as a curve $(\mathbf{x}(\cdot), \theta(\cdot)) : [0, \infty) \to \Omega \times [0, 2\pi)$ parametrized by arc-length in Ω:

$$|\dot{\mathbf{x}}(s)| = 1, \quad s > 0. \tag{2}$$

Note the relations

$$\theta(s) = \tan^{-1}(\dot{y}(s)/\dot{x}(s)), \tag{3}$$

$$\dot{\mathbf{x}}(s) = (\cos(\theta(s)), \sin(\theta(s))), \tag{4}$$

where ever $\mathbf{x}(\cdot)$ is smooth. We say that a path is *feasible* if it is contained in $\Omega_{free} \times [0, 2\pi)$. The *average curvature* between two parametrization variables $s_1, s_2 > 0$ of a path $(\mathbf{x}(\cdot), \theta(\cdot))$ is defined as $AC_{\mathbf{x}}(s_1, s_2) = |\theta(s_2) - \theta(s_1)|/|s_1 - s_2|$.

Throughout this article, we assume that the vehicle travels with unit speed (which is automatically implied by (2)) and has a turning angle constraint that induces the constraint $AC_{\mathbf{x}} \geq \rho_{\min} > 0$, where $\rho_{\min}$ is the *minimum turning radius*. It is easy to see that if $\theta(\cdot)$ is differentiable, the constraint is equivalent to

$$|\dot{\theta}(s)| \leq \rho_{\min}^{-1}, \quad s > 0. \tag{5}$$

Denote $\mathcal{A}_{\rho_{\min}, \mathbf{x}_0, \theta_0}$ the set of *admissible path from the pose* $(\mathbf{x}_0, \theta_0)$, that is, feasible paths that satisfy (2), (5) and $(\mathbf{x}(0), \theta(0)) = (\mathbf{x}_0, \theta_0)$. Such paths are precisely the trajectories generated by the Dubins' car among impenetrable obstacles.

Finally, given a target point $\mathbf{x}_f$ we define the corresponding *value function* $u : \Omega \times [0, 2\pi) \to \mathbb{R}^+ \cup \{0\}$:

$$u(\mathbf{x}, \phi) = \inf\{t : (\mathbf{x}(\cdot), \theta(\cdot)) \in \mathcal{A}_{\rho_{\min}, \mathbf{x}, \phi}, \mathbf{x}(t) = \mathbf{x}_f\}. \tag{6}$$

In case the final orientation of the vehicle is specified, there is an additional condition $\theta(t) = \theta_f$. In other words, the function u is the optimal cost-to-go for the Dubins' car problem with given constraints, an initial pose, and a target position (or a pose).

B. Formal Derivation of the Hamilton-Jacobi Equation

Using (6), we start with the dynamic programming principle for our problem:

$$u(\mathbf{x}, \phi) = \inf\{u(\mathbf{x}(\Delta t), \theta(\Delta t)) + \Delta t : (\mathbf{x}(\cdot), \theta(\cdot)) \in \mathcal{A}_{\rho_{\min}, \mathbf{x}, \phi}\}. \tag{7}$$

Rearranging the terms, dividing by Δt, and taking $\Delta t \to 0$, we have,

$$-1 = \inf\{\nabla u \cdot (\dot{\mathbf{x}}, \dot{\theta}) : |\dot{\mathbf{x}}| = 1, |\dot{\theta}| \leq \rho_{\min}^{-1}\}. \tag{8}$$

Furthermore, by applying (4) we obtain the *Hamilton-Jacobi-Bellman equation* [19]:

$$-1 = \cos(\theta)u_x + \sin(\theta)u_y + \inf_{|\dot{\theta}|\leq\rho_{min}^{-1}}\{\dot{\theta}u_\theta\}, \quad (9)$$

where subscripts denote partial derivatives. Finally, the infimum in the last term can be eliminated by assuming the bang-bang principle

$$\dot{\theta} = \pm\rho_{\min}^{-1}. \quad (10)$$

Thus, we arrive at the HJ equation

$$-1 = \cos(\theta)u_x + \sin(\theta)u_y - |u_\theta|/\rho_{\min}. \quad (11)$$

Since the target has zero cost-to-go, we have the following boundary conditions: $u(\mathbf{x}_f, \theta_f) = 0$, or $u(\mathbf{x}_f, \theta) = 0$ for $\theta \in [0, 2\pi)$, if no final orientation is given. The obstacles can be interpreted as the locations of infinite cost-to-go: $u(\mathbf{x}, \theta) = \infty$ for $\mathbf{x} \in \Omega_{obs}, \theta \in [0, 2\pi)$.

It can be shown that the bang-bang control (10) does not characterize optimal trajectories if the obstacle boundaries have curvature between 0 and 1 [9]. If such obstacle boundaries exist, the bang-bang control will induce infinite switching of controls to trace the boundaries. Such a phenomenon is referred to as *chattering control* [19]. In a numerical setting, the number of switchings is $O(1/h)$, where h is the grid refinement. As $h \to 0$, the resulting numerical path approaches the true optimal path, even when (10) is implemented.

C. Numerical Implementation

To solve (11), we propose an upwind, monotone finite difference discretization, and a fast sweeping update scheme. Set up a three dimensional uniform Cartesian grid with refinement (h_x, h_y, h_θ). Let $u_{i,j,k} \approx u(ih_x, jh_y, kh_\theta)$ be the approximation of the solution at the grid nodes. Denote $\xi_k = \text{sgn}(\cos(\theta_k))$ and $\nu_k = \text{sgn}(\sin(\theta_k))$. We approximate the derivatives by upwind discretizations

$$(\cos(\theta)u_x)_{i,j,k} = -|\cos(\theta_k)|\frac{u_{i+\xi_k,j,k} - u_{i,j,k}}{h_x}, \quad (12)$$

$$(\sin(\theta)u_x)_{i,j,k} = -|\sin(\theta_k)|\frac{u_{i,j+\nu_k,k} - u_{i,j,k}}{h_x}, \quad (13)$$

and a monotone discretization [28]

$$\left(\frac{|u_\theta|}{\rho_{\min}}\right)_{i,j,k} = \max\left\{\frac{u_{i,j,k+1} - u_{ijk}}{\rho_{\min}h_\theta}, \frac{u_{i,j,k-1} - u_{i,j,k}}{\rho_{\min}h_\theta}, 0\right\}. \quad (14)$$

For simplicity, we set $h_x = h_y = h$. Substituting (12), (13), and (14) into (11), and solving for u_{ijk}, we have

$$\begin{aligned} u^*_{i,j,k} &= \{|\cos(\theta_k)|u_{i+\xi_k,j,k} + |\sin(\theta_k)|u_{i,j+\nu_k,k} \\ &+ h\max\{u_{i,j,k\pm1}\}/(\rho_{\min}h_\theta) - h\} \\ &/ \{|\cos(\theta_k)| + |\sin(\theta_k)| + h/(\rho_{\min}h_\theta)\}, \end{aligned} \quad (15)$$

if $(|u_\theta|/\rho_{\min})_{ijk} \neq 0$, and

$$\begin{aligned} u^{**}_{i,j,k} &= \{|\cos(\theta_k)|u_{i+\xi_k,j,k} + |\sin(\theta_k)|u_{i,j+\nu_k,k} - h\} \\ &/ \{|\cos(\theta_k)| + |\sin(\theta_k)|\}, \end{aligned} \quad (16)$$

if $(|u_\theta|/\rho_{\min})_{i,j,k} = 0$.

The boundary conditions are implemented by setting $u_{i,j,k} = 0$ at the nearest nodes to $\mathbf{x}_f$ (and θ_f for a final orientation constraint) and $u_{i,j,k} = \infty$ at nodes inside Ω_{obs}. The boundary of the domain is also set to infinity.

To solve the system of nonlinear equations (15), (16), we apply the fast sweeping algorithm [24]. This involves the update scheme (denoting the iteration by superscripts)

$$u^{n+1}_{i,j,k} \leftarrow \min\{u^{*n}_{i,j,k}, u^{**n}_{i,j,k}, u^n_{i,j,k}\} \quad (17)$$

to be iterated by the Gauss-Seidel method. Iterations are carried out by 'sweeping' in eight directions: increasing and decreasing in each i, j, and k.

The nodes used to approximate (11) locally around a particular node (the *stencil*) used in (15) and (16) are shown in Fig. 2(a). To approximate the partial derivative in x and y directions more accurately, we can use a larger stencil. For example, we can add the nodes $\{(i \pm 1, j \pm 1, k)\}$ to the stencil (shown in Fig. 2(b)). To this end, we first use these nodes to approximate $u_{\bar{x}}, u_{\bar{y}}$, where $(\bar{x}, \bar{y})$ is a the usual cartesian coordinates rotated 45 degrees clockwise. Next, use the identities, $u_x = (u_{\bar{x}} + u_{\bar{y}})/\sqrt{2}$, $u_y = (-u_{\bar{x}} + u_{\bar{y}})/\sqrt{2}$ to derive similar formulae for (15) and (16); denote the corresponding left hand side as w^*_{ijk} and w^{**}_{ijk}, respectively. Then, the update scheme analogous to (17) becomes $u^{n+1}_{ijk} \leftarrow \min\{u^{*n}_{i,j,k}, u^{**n}_{i,j,k}, w^{*n}_{i,j,k}, w^{**n}_{i,j,k}, u^n_{i,j,k}\}$. Similarly, one can create arbitrary accurate and wide stencils; for our computations, we used a 16-neighbor stencil as shown in Fig. 2(c).

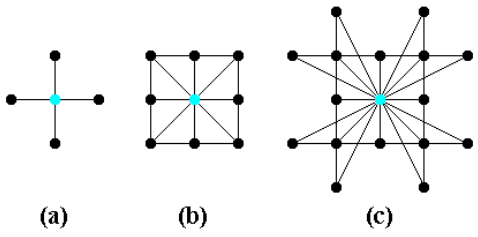

Fig. 2. Different choices of stencils. The blue dots are the nodes (i, j).

D. Computing Trajectories

Once a numerical approximation $u_{i,j,k}$ to the value function is obtained, optimal paths from a pose $(\mathbf{x}_0, \theta_0) \in \Omega_{free} \times [0, 2\pi)$ can be computed efficiently by tracing the characteristic curves of (11) back to the target location $\mathbf{x}_f$ or pose $(\mathbf{x}_f, \theta_f)$. The corresponding dynamical system is

$$\dot{\mathbf{x}}(s) = -(\cos(\theta(s)), \sin(\theta(s))) \quad (18)$$

$$\dot{\theta}(s) = -\text{sgn}(u_\theta(\mathbf{x}(s), \theta(s)))/\rho_{\min}. \quad (19)$$

We use a forward Euler scheme to solve this numerically, with a centered difference approximation for the partial derivative in (19). The values of u not on the nodes are approximated by a nearest-neighbor interpolation.

E. Numerical Results

For all computations in this section we use $\Omega = [-1, 1]^2$ discretized uniformly by a 200×200 grid. The θ values are discretized with 200 uniform nodes. We use $\rho_{\min} = 0.2358$. The time step used for the forward Euler scheme in computing the trajectories was set to the grid refinement $h = 0.01$. Typically about 5 to 7 iterations (each iteration involves 8 sweeps) were sufficient for convergence, depending on the domain shape.

Fig. 3 shows the 0.5 and 0.9 level sets of the value function $u_{i,j,k}$, with $\mathbf{x}_f = (0, 0)$ and no final orientation constraint. No obstacles are present in this example for simplicity of presentation. The level sets of $u_{i,j,k}$ define the equal cost-to-go poses to the target. Note that the turning radius constraint is noticeable near the central axis in the 0.5 level set; for locations of larger cost-to-go, this becomes less evident.

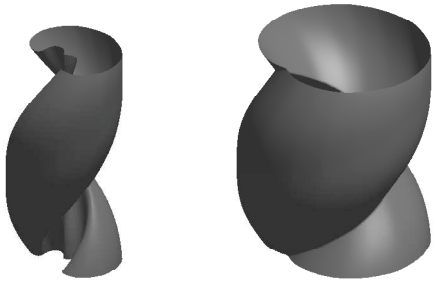

Fig. 3. Left to right: the 0.5 and 0.9 level sets of (u_{ijk}), the value function (6), for the case $\mathbf{x}_f = (0, 0)$ and no final orientation constraint. The vertical axis represents the θ direction.

Next, we consider the case where obstacles are present. We make the numerical obstacles slightly larger than the physical obstacle to take into consideration the size of the car (recall that numerically, the car position is represented by a single point). We will call the difference of these two sets, the *buffer region*.

In Fig. 4 we show two optimal paths, one with no constraint on θ_f, and another with $\theta_f = \pi/4$. Both have the same initial pose.

III. A Practical Algorithm for Steering a Simple Car

We now use the formulation proposed in section II to address the realistic navigation scenario for a (reversible) simple car, where smooth acceleration, inertia, rough surfaces, and mechanical imprecisions must be taken into consideration.

The main difficulty is an autonomous vehicle may stray off from a given optimal path. In the case of a Dubin's car, a small offset from the optimal path may result in a catastrophic change of strategy [1, Chapter 13]. For example, in a tight domain, a small perturbation can steer the vehicle from an initial pose with an admissible path to a pose from which it can no longer reach the target, i.e. finite to infinite cost-to-go.

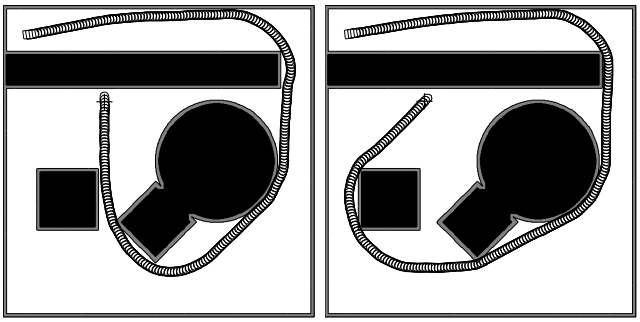

Fig. 4. Optimal paths, among obstacles. The target (same for both cases) is shown by a '+', the obstacles by black blobs and the buffer region in gray. Left: no final orientation constraint, path length is 4.9648. Right: with final orientation constraint $\theta_f = \pi/4$, path length is 5.5980.

In contrast, the option of reversibility can alleviate such a difficulty. Below, we introduce a scheme for navigating a reversible simple car that uses the HJ formulation, and demonstrate its performance in a robot testbed environment.

A. A Semi Real-time Correcting Scheme

We first propose a scheme that adjusts the optimal path of the vehicle as it deviates from its originally optimal path due to realistic factors.

Initially, set $m = 0$,

Step 1. compute the optimal path $\mathcal{P}_m \subset \Omega_{free}$ from the pose $(\mathbf{x}_m, \theta_m)$ to the target point $\mathbf{x}_f$ (or pose $(\mathbf{x}_f, \theta_f)$),
Step 2. steer the vehicle for a 'short distance' along $\mathcal{P}_m$,
Step 3. record the vehicle pose as $(\mathbf{x}_{m+1}, \theta_{m+1})$,
Step 4. let $m \leftarrow m + 1$ and repeat from step 1.

We shall call this the semi real-time correcting (SRC) scheme. Note that in step 1, one only needs to compute $\mathcal{P}_m$ that is necessary for step 2. We assume a mechanism to track the current pose of the vehicle (step 3).

Next, to add the option of reversiblity, we add another candidate path in step 1 of the SRC scheme:

Step 1. (modified) Compute the optimal path $\mathcal{P}_m^1 \subset \Omega_{free}$ from pose $(\mathbf{x}_m, \theta_m)$, and another optimal path $\mathcal{P}_m^2 \subset \Omega_{free}$ from the the pose $(\mathbf{x}_m, \theta_m + \pi)$. Let $\mathcal{P}_m$ be the shorter of $\mathcal{P}_m^i, i = 1, 2$.

Although one may be tempted to believe that this solves an approximation to the Reeds-Shepp's car problem [3], it does not. Rather, we are solving a Dubins' car problem from each recorded vehicle pose $(\mathbf{x}_m, \theta_m)$, allowing the vehicle to start forward or backwards.

B. HJ-ased SRC Scheme

The HJ approach to optimal path planning for the Dubins' car problem computes optimal trajectories efficiently if the value function is provided. Indeed, the value function contains information of *all* optimal paths to the target location $\mathbf{x}_f$ (or pose $(\mathbf{x}_f, \theta_f)$). Consequently, the bulk of the computational cost is in computing the value function.

To exploit the full advantage of the HJ formulation, we consider the following situation: suppose that the map Ω, the

target location $\mathbf{x}_f \in \Omega_{free}$, and possibly the final orientation $\theta_f \in [0, 2\pi)$ are provided. Then, the corresponding value function $u_{i,j,k}$ can be computed offline prior to steering the vehicle. We outline the HJ based SRC scheme: given a map $\Omega = \Omega_{free} \cup \Omega_{obs}$, a minimum turning radius $\rho_{\min}$, and a desired target location $\mathbf{x}_f$ (or pose $(\mathbf{x}_f, \theta_f)$),

1) precompute the value function u_{ijk} (section II-C),
2) perform the SRC scheme (section III-A); use the method in section II-D to compute the paths $\mathcal{P}_m$.

Strictly speaking, since the turning radius of a simple car differs for forward and backward motion, one must precompute two value function. However, we have found that the algorithm performs well in practice by using only the value function of the larger turning radius.

C. Experimental Results

We validated the HJ based SRC scheme on an autonomous simple car, using the testbed introduced in [29]. In [30] our scheme is applied to steer an autonomous vehicle while it searches for a signal source in a bounded domain with obstacles.

We used the same map as in section II-E including the buffer region, but scaled to a 140×140 cm square. The car size was $7 \times 3.8 \times 4.6$ cm (L×W×H) and had a minimum turning radius of $\rho_{\min} = 16.51$ cm. The value function was computed on a $200 \times 200 \times 200$ grid. The car location was sampled and recorded approximately every 15 cm travelled; this was based on the smallest distance the car can maneuver with reasonable accuracy.

Fig. 5 illustrates the SRC scheme (without the reversibility option) for a final target location $\mathbf{x} = (45.5, 98)$. Notice how the vehicle is able to stay in the vicinity of the the optimal path even when it strays off. Fig. 6 shows the optimal path under the same parameters with the reversibility option. Fig. 7 shows a sequence of intermediate steps illustrating how the SRC scheme with reversibility plans its next motion. For this example, the reversibility option generates a path shorter by about 20%. Finally, Fig. 8 shows the case when an additional final orientation constraint $\theta_f = \pi/4$ is added.

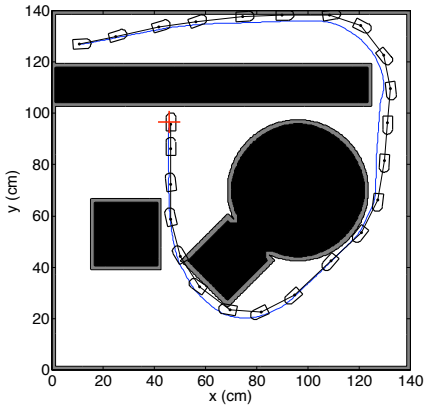

Fig. 5. The path taken by the car steered by the SRC scheme. The blue curve marks $\mathcal{P}_0$, the optimal path from the initial pose. The target is shown by a red '+'. Length of piecewise linear car path is 348 cm.

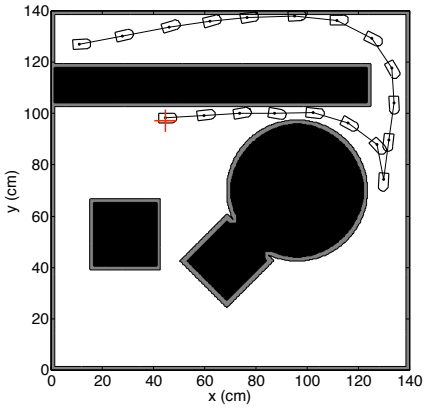

Fig. 6. The path taken by the car steered by the SRC scheme with the reversibility option. The target is shown by a red '+'. Length of piecewise-linear car path is 280 cm.

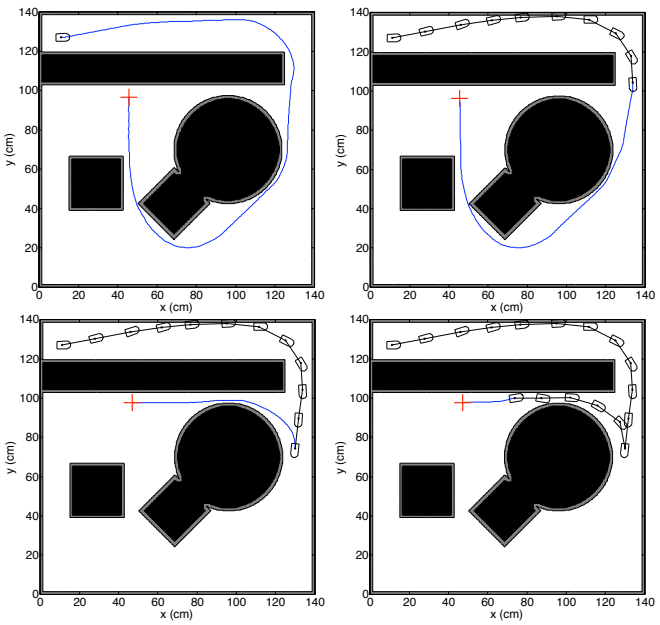

Fig. 7. Intermediate steps of the SRC scheme with the reversibility option. The blue curve marks $\mathcal{P}_m$, the optimal path emanating from the last recorded car location, and the red '+' shows the target $\mathbf{x}_f$. Note how the scheme initially generates the Dubins' car path, then reverses its direction when the car is able to steer through the small opening above the circular obstacle.

IV. Conclusions

In this article, we first formulated a Hamilton-Jacobi (HJ) equation to solve the problem of computing an optimal path, given an initial pose and a target pose, under a curvature constraint. This models a reversible simple car with a turning radius bounded from below. We provided a numerical method to solve this problem.

In the second part we applied our HJ formulation to a general scheme to steer an reversible simple car. This scheme itself can be worked in tandem with other optimal path planning algorithms. The performance of this scheme was experimentally validated. We refer the reader to a future publication for a rigorous analysis of the proposed

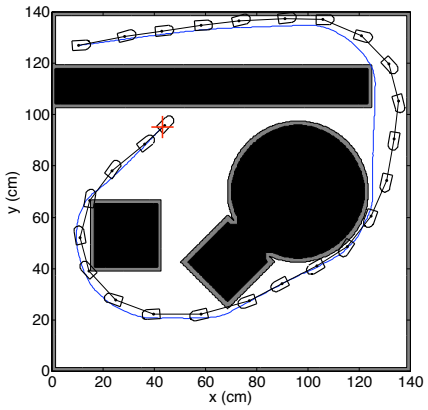

Fig. 8. The same scheme as Fig. 6, but with a final orientation constraint $\theta_f = \pi/4$. The blue curve marks $\mathcal{P}_0$, the optimal path from the initial pose. The target is shown by a red '+'. Length of piecewise linear car path is 395 cm.

HJ formulation and its numerical discretization.

V. Acknowledgements

We would like to thank professor Andrea Bertozzi for her suggestions and for providing us with the robotic testbed to conduct our experiments in the UCLA Applied Mathematics Laboratory funded by an ARO MURI grant 50363-MA-MUR. We would also like to thank professors Stanley Osher and Alexander Vladimirsky for their valuable advice.

Landa's, Takei's, and Tsai's research is supported by a MURI subcontract from U. of South Carolina and an NSF Algorithm for Threat Detection Grant DMS-0914465. Shen's research is supported by an ARO MURI grant 50363-MA-MUR.

References

[1] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006

[2] L. E. Dubins. On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents. American Journal of Mathematics, vol. 79, no. 3, Jul. 1957, pp. 497-516.

[3] J. A. Reeds and L. A. Shepp. Optimal paths for a car that goes both forwards and backwards. Pacific J. Math., 145 (1990), pp. 367393.

[4] S. Fortune and G. Wilfong, Planning constrained motion. Annals of Math. and Air, 3:21-82, 1991.

[5] H.-K. Ahn, O. Cheong, J. Matousek and A. Vigneron. Reachability by paths of bounded curvature in convex polygons. Annual Symposium on Computational Geometry, Proceedings of the sixteenth annual symposium on Computational geometry, Clear Water Bay, Kowloon, Hong Kong, 2000, pp. 251 - 259

[6] J. Barraquand and J. C. Latombe, Nonholonomic multibody mobile robots: Control lability and motion planning in the presence of obstacles, Algorithmica, 10 (1993), pp. 121155.

[7] H. Wang and P. K. Agarwal. Approximation algorithms for curvature-constrained shortest paths. in Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, Atlanta, GA, 1996, pp. 409418.

[8] P. K. Agarwal, P. Raghavan, and H. Tamaki. Motion planning for a steering-constrained robot through moderate obstacles. In *Proc. Annual Symposium Theory Computing*, 27, pp. 343352, 1997.

[9] J.-D. Boissonnat and S. Lazard. A polynomial-time algorithm for computing a shortest path of bounded curvature amidst moderate obstacles. In Proceedings ACM Symposium on Computational Geometry, pp. 242-251, 1996

[10] P. Jacobs and J. Canny. Planning smooth paths for mobile robots, in Nonholonomic Motion Planning. Kluwer Academic Publishers, Norwell, MA, 1992

[11] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, Vol. I, 2nd Ed. Athena Scientific, Belmont, MA, 2001.

[12] J. Reif and H. Wang. Nonuniform Discretization for Kinodynamic Motion Planning and its Applications. SIAM J. Comput., Vol. 30, No. 1, pp. 161190

[13] J.-H. Lee, O. Cheong, W.-C. Kwon, S.-Y. Shin and K.-Y. Chwa. Approximation of Curvature-Constrained Shortest Paths through a Sequence of Points. Lecture Notes In Computer Science; Vol. 1879, Proceedings of the 8th Annual European Symposium on Algorithms, 2000, pp. 314- 325

[14] J. P. Laumond, P. E. Jacobs, M. Taix, and R. M. Murray. A motion planner for nonholonomic mobile robots. IEEE Trans. Robotics and Automation, 10 (1994), pp. 577593.

[15] P. Konkimalla and S. M. Lavalle. Efficient Computation of Optimal Navigation Functions for Nonholonomic Planning. In Proc. First IEEE Intl Workshop on Robot Motion and Control, 1999.

[16] L. C. Evans. *Partial Differential Equations*. Graduate Studies in Mathematics, Vol 19, AMS, Providence, Rhode Island, 1998.

[17] R. E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[18] M. Bardi. Some Application of Viscosity Solutions to Optimal Control and Differential Games. In *Lecture Notes in Mathematics, Viscosity Solutions and Applications*. Editors: I. Capuzzo-Dolcetta, P.-L. Lions. Springer, 1997.

[19] M. Bardi and I. Capuzzo-Dolcetta. *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations*. Birkhauser, 1997.

[20] M. Falcone and R. Ferretti. Semi-Lagrangian schemes for Hamilton-Jacobi equations, discrete representation formulae and Godunov methods. J. of Comp. Physics, Vol. 175 (2), 2002, pp. 559-575

[21] R. Kimmel and J. A. Sethian. Optimal algorithm for shape from shading and path planning. Journal of Mathematical Imaging and Vision, vol. 14, no. 3, pp. 237244, 2001.

[22] J. N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. IEEE Transactions on Automatic Control, vol. AC-40, no. 9, pp. 15281538, 1995.

[23] C.-Y. Kao, S. Osher and Y.-H. Tsai. Fast sweeping methods for static Hamilton-Jacobi equations. SIAM Journal on Numerical Analysis, vol. 42, no. 6, pp. 26122632, 2005.

[24] Y.R. Tsai, L.T. Cheng, S. Osher and H.K. Zhao. Fast Sweeping Algorithms for a Class of Hamilton-Jacobi Equation. SIAM Journal on Numerical Analysis, Vol 41, No 2, 2003, pp.673-694.

[25] K. Alton and I. Mitchell. Optimal Path Planning under Different Norms in Continuous State Spaces. Proceedings of the IEEE International Conference on Robotics and Automation, pp. 866-872 (May 2006).

[26] J.A. Sethian and A. Vladimirsky. Ordered upwind methods for static Hamilton-Jacobi equations. Proc. Natl. Acad. Sci. USA 98/20: 11069-11074 (2001).

[27] C.-Y. Kao, S. Osher and J. Qian. Lax-Friedrichs Sweeping Scheme for Static Hamilton-Jacobi Equations. UCLA Math CAM Report 03-38, Journal of Computational Physics,196(1), Pages 367-391,2004

[28] A. Oberman. Convergent Difference Schemes for Nonlinear Elliptic and Parabolic Equations: Hamilton-Jacobi Equations and Free Boundary Problem. SIAM Journal on Numerical Analysis, Vol 44 (2006) No. 2 pp. 879-895.

[29] K. K. Leung, C. H. Hsieh, Y. R. Huang, A. Joshi, V. Voroninski, and A. L. Bertozzi. A second generation micro-vehicle testbed for cooperative control and sensing strategies. In American Control Conference, 2007. ACC07, pages 19001907, 2007.

[30] Y. Landa, H. Shen, R. Takei and R. Tsai. Autonomous Source Discovery and Navigation in Complicated Environments. preprint.

[31] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki and S. Thurn. *Principles of Robot Motion*. MIT Press, 2005.

[32] J. P. Laumond and J. J. Risler. Nonholonomic systems : controllability and complexity. Theoretical Computer Sciences, 157, pp. 101–114, 1996.

[33] S. M. LaValle. A Game Theoretic Framework for Robot Motion Planning. PhD thesis. University of Illinois, Urbana, IL, 1995.

[34] S. Osher and C.-W. Shu. High-order essentially nonoscillatory schemes for HamiltonJacobi equations. SIAM J. Numer. Anal., 28 (1991), pp. 907922.